Debreceni Egyetem
Informatikai Kar

# A Huygens-Fresnel elv számítógépes szimulációja középiskolákban

Témavezető:
Dr. Papp Zoltán

Készítette:
Sztojka Ferenc

Debrecen
2007

*Ezúton is köszönöm Dr. Papp Zoltán*
*tanár úrnak mindazt a segítséget, amit*
*a három év során nyújtott*

# *Tartalomjegyzék*

## Bevezetés

A középiskolában kísérleti eszköz hiánya miatt elég nehezen megvalósítható a különböző hullámjelenségek bemutatása és magyarázata. Az egyik legnagyobb probléma a Hugens-Fresnel elv szemléltetése. A Hugens-Fresnel elv a hullám terjedésére ad egy magyarázatot. Azt mondja ki, hogy a hullámfront minden egyes pontja elemi hullámok kiinduló pontja és a következő hullám frontot ezen elemi hullámok interferenciája adja. Ez azt jelenti, hogy a hullámtér egy pontjának aktuális állapotának meghatározásához, ki kel számítani a hullámfront minden pontja esetén, hogy ha csak ő lenne hatással a hullámtér adott pontjára, hogy milyen hatása lenne a hullámtér adott pontjára. Majd ezeket a hatásokat összegezve meg kapjuk a hullámtér adott pontjának állapotát. Ez, nagy „felületű" hullámfront esetén, nagyon sok számolást jelent. Tovább növeli a számolás mennyiségét, látványos hatás elérése érdekében, a hullámtér nagysága. Ezért a különböző hullám jelenségek Hugens-Fresnel elvvel való szemléltetésére a számítógépes szimuláció tűnik megfelelőnek.

### *Miért a pascal nyelv?*

A szimulációs program megírását nagyban nehezíti, hogy a hullámtér minden pontjának állapotát külön-külön meg kell tudni adni az esetleg több hullám centrumhoz képest. Ez azt jelenti, hogy ha a képernyő képpontjaira akarjuk leképezni a hullámmozgást, akkor a képernyő pixeleit külön-külön kell tudni kezelni ez 640x480-as felbontás esetén 307200 darab pixel beállítást jelent. Az alap programozási nyelvek alap grafikai moduljai ezt olyan lassan végzik el hogy arra normális animációt nem lehet építeni. Kísérleteztem OpenGL-el is. Sajnos az OpenGL inkább poligonok térbeli műveleteire van optimalizálva, a pixel műveletek 640x480-as méretben olyan lassúnak bizonyultak, hogy animációra alkalmatlanak bizonyult. Szerencsére az Interneten találtam a Free Pascal-hoz egy olyan, VGFX nevezetű 32bit-es grafikai modult, ami megfelelőnek tűnt a feladat megoldásához. (http://codexonline.hu/CodeX10//alap/freepascal/InczeAttila/part1.htm 2007 április 18.-án még elérhető)A VGFX képernyő-kezelési módszerként a DirectDraw-ot használja, ezért csak Windows alá jó.

## A körhullámok szimulációja

A szimulációhoz a hullámtér minden egyes pontjának állapotát külön-külön meg kell tudni adni, az esetleg több, hullámcentrum állapotának függvényében. A hullámtér egy pontjának aktuális állapotát az $y = A'\sin(\frac{2\cdot\pi}{T}\cdot t - \frac{2\cdot\pi}{\lambda}\cdot x)$ hullámfüggvény írja le. Ahol T a periódusidő, λ a hullámhossz, t az idő, x pedig a hullámforrástól való távolság A hullámforrás energiája egyenesen arányos a hullámforrás A amplitúdójának a négyzetével. Kör hullámokat feltételezve, ez az energia oszlik szét a kör alakú hullámfront teljes kerületén. Ez azt jelenti, hogy $A' \sim \frac{A}{\sqrt{2\cdot\pi\cdot x}}$. Ez, egy hullámcentrum, és 640x480-as felbontás esetén 307200 számítást jelent. A hullámtér pontjait a 640x480 felbontású képernyő pontjaival szemléltetjük. A hullámtér egy adott pontjának aktuális állapotát színárnyalatokkal. Mivel a VGFX a vörösre 0-32 egészértéket, a zöldre 0-64 egészértéket, a kékre 0-32 egészértéket engedélyez, a hullámtér egy pontjának állapotát, a finomabb léptetési lehetőségek miatt, a zöld árnyalataival fogjuk szemléltetni. A hullámtérnek azt a pontját, amely éppen egy hullámhegy halad át, világosabb színnel, amelyen egy hullám völgy halad át, sötétebb színnel szemléltetjük. Mivel a hullámtér egy adott pontja $y = A'\sin(\frac{2\cdot\pi}{T}\cdot t - \frac{2\cdot\pi}{\lambda}\cdot x)$ hullámfüggvény szerinti mozgást végez, szükség van a hullámtér adott pontjának és a hullámforrás távolságának kiszámolására. Azért, hogy a program, futás közben minél kevesebbet számoljon, célszerű ezt előre kiszámolni és egy 640x480-as tömbben lementeni egy fájlba. (tavolsag.kep) Annak érdekében, hogy a hullám forrás helyzetét megváltoztathatjuk, (-320;-240),(960; 720) virtuális hullámtér távolság adatait kéne lementeni. Mivel azonban a (320; 240) koordinátájú hullámforráshoz képest a távolság adatok x=320, és y=240 egyenesekre nézve tengelyesen szimmetrikusan helyezkednek el, mégis csak elegendő egy 640x480-as integer típusú tömbben lementeni a távolság adatokat. A hullámtér egy pontjai színkódjainak kiszámítása szintén nagyon idő igényes számolás ezért ezt a műveletet is célszerű előre kiszámítani és lementeni. Hely takarékosság igénye miatt a érdemes, egy 801x124-es tömb oszlopvektoraiba az oszlop indexnek megfelelő, x távolságértékkel kiszámolni egy teljes periódus, egészekre kerekített értékeit, majd a tömböt szintén lementeni egy fájlba. (ritmus.kep) A megfelelő program részlet:

```
program metes;
uses crt,Windows ;
type
tavolsag=array[-320..320,-240..240]  of integer;
```

```
ritmus=array[0..800] of array[0..123]of shortint;
var
i,j,k:integer;
tav: file of tavolsag;
utem:file of  ritmus;
negyed_tav :  array[-320..320,-240..240]  of integer;
teljes_ritmus : array[0..800] of array [0..123] of shortint;
begin
assign(tav,'C:\Szakdolg\metes\adatok\tavolsag.kep');
assign(utem,'C:\Szakdolg\metes\adatok\ritmus.kep');
rewrite(tav);
rewrite(utem);
for i:=-320 to 320 do
 for j:=-240to 240 do
 negyed_tav[i][j]:=round(sqrt(sqr(320-i)+sqr(240-j)));
for i:=0 to 800 do
 for k:=0 to 123 do
 if i>0 then
 teljes_ritmus[i][k]:=round((31/sqrt(2*Pi*i))*sin((Pi*k/62)-
Pi*i/10))
 else
  teljes_ritmus[i][k]:=round(31*sin((Pi*k/62)-Pi*i/10));
write(tav,negyed_tav);
write(utem,teljes_ritmus);
close(tav);
close(utem);
end.
```

A program, futás közben csak azt nézi meg, hogy a hullámtér vizsgált pontja az aktuális t pillanatban milyen x távolságra van a hullámforrástól, és kiolvassa a 801x124-es tömb x-edik sorának (t mod 124)-edik értékét. Mivel a $y = A'\sin(\frac{2\cdot\pi}{T}\cdot t - \frac{2\cdot\pi}{\lambda}\cdot x)$ függvény negatív értékeket is ad, a színkódok pedig nem negatívak, még hozzáadunk 31-et, így megkapjuk a hullámtér vizsgált pontját jellemző színkódot.

## Két hullámforrás

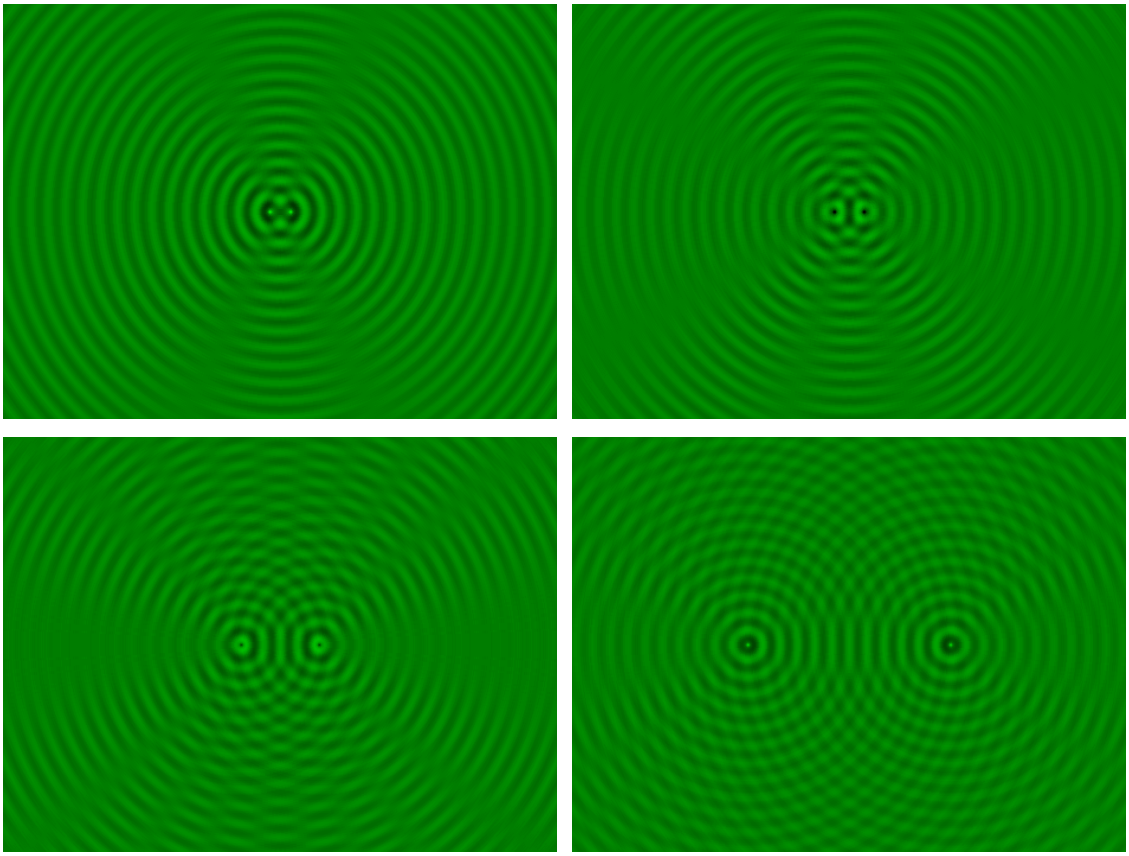
Eredetileg a tavolsag.kep fájlban a hullámtér pontjainak a (320;240) koordinátájú hullámforrástól való távolság adatok vannak lementve. A hullámforrás (v;u)koordinátájú vektorral való 'elmozdítását' úgy érhetjük el, hogy amikor a hullámtér egy P(x,y) koordinátájú pontjának keressük a távolságát az elmozdított hullámcentrumhoz képest, akkor a P(x-v;y-u) pontnak olvassuk ki a távolságát a (320;240) koordinátájú hullámcentrumhoz képest. A programban így két hullámforrást kezelünk. A hullámtér egy pontjának állapotát e két hullámcentrum által okozott állapotok összege adja. A kiszámított állapot egészre kerekített értékéhez még hozzáadva 31-et megkapjuk, az adott pont színkódját. A megfelelő program részlet:

```
program hullamok;
uses
crt,
Windows,
VGFX,
VGFX_2D,
VGFX_Sprites,
VGFX_Text,
VGFX_win32,
VGFX_Files,
VGFX_Errors,
BigFile2,
SysUtils;
type
matrix=array[-320..320,-240..240]of integer;
ritmus=array[0..800]of array[0..128]of real;
var
vw1,vw2 : virtualwindow;
tav:file of matrix;
utem:file of ritmus;
tavolsag:array[-320..960,-240..720]of integer;
hullamter:array[0..128]of array[-320..960,-240..720]of
shortint;
utemek: array[0..800]of array[0..128]of real;
n,i,j,v,k:integer;
```

```
procedure kezdoallapot;
 var
 negyed_tav:matrix;
 begin
 assign(tav,'tavolsag.kep');
 assign(utem,'ritmus.kep');
 reset(tav);
 reset(utem);
 read(tav,negyed_tav);
 read(utem,utemek);
 for i:=-320 to 320 do
  for j:=-240 to 240 do
  begin
  tavolsag[i][j]:= negyed_tav[i][j];
  tavolsag[640-i][j]:= negyed_tav[i][j];
  tavolsag[i][480-j]:= negyed_tav[i][j];
  tavolsag[640-i][480-j]:= negyed_tav[i][j];
  end;
  close(tav);
  close(utem);
 end;
 procedure inditas;
 begin
 for k:=0to 128 do
  for i:=-320 to 960 do
   for j:=0 to 480 do
   hullamter[k][i][j]:=round(utemek[tavolsag[i][j]][k]);
  end;
begin
v:=0;
k:=0;
n:=1;
kezdoallapot;
inditas;
```

```
Window_RegisterClass(WIN_Normal);
Window_Main_Handle:= Window_CreateClass('NewTime
v.2.0Win',WIN_Normal);
CheckWMCreate;
init_graph(640,480);
init_vw(vw1,640,480,true);
init_vw(vw2,640,480,true);
REPEAT
flip_vw(vw2.vwoffset,vw1.vwoffset);
if ktaste[37] and (v<320)then v:=v+1;
if ktaste[39] and (v>0)then v:=v-1;
for i:=0 to 640 do
 for j:=0 to 480 do
 begin
 if ktaste[40] then
putpixel(vw2,i,j,0,abs(hullamter[k][i+v][j])+abs(hullamter[k][
i-v][j]),0)
 else
 putpixel(vw2,i,j,0,hullamter[k][i+v][j]+hullamter[k][i-
v][j]+31,0) ;
 end;
k:=(k+5)mod 128;
Flip_DD(vw1.VWOffset);
GETALLMESSAGES;
HANDLEMINIMIZED;
UNTIL ktaste[27];
kill_vw(vw1);
kill_vw(vw2);
kill_graph;
end.
```

A program futtatásával létrehozott szimuláció alkalmasnak tűnik az inteferencia jelenségek magyarázatához. Mint ahogy az alábbi ábrák mutatják tisztán, és jól láthatóan kirajzolódtak az interferenciagörbék. A hullámcentrumok mozgatásával meg lehet figyelni ezen interferencia-görbék változását.



## Hullám elhajlása résen

Ha hullám résen halad át, akkor a rés méretétől függően, azt tapasztaljuk, hogy behatol az 'árnyékos' területre. Ez az árnyékos részekbe való bejutás annál jellemzőbb minél kisseb a rés mérete a hullámhosszúsághoz képpest. A magyarázat a Hugens-Fresnel elvből adódik. E szerint tekintsük úgy a rést, mintha a rés pontjaiból elemi hullámok indulnának ki. A résmögötti hullámtér pontjait, pedig ezen elemi hullámok interferenciája határozná meg. Ekkor az árnyékos részek, úgy alakulnak ki, hogy az ilyen pontokban ezen elemi hullámok éppen kioltják egymást. A szimuláció során a résre egyenes vonal hullámokat küldünk, hogy a rés pontjai azonos fázisban legyenek. A program futásának felgyorsítása érdekében a nagyobb mennyiségű számításokat itt is előre elvégeztük és lementettük egy fájlba (ritmus1.kep). A megfelelő program részlet:

```
program metes;
uses crt,Windows ;
type
ritmus=array[0..320]of array[0..123]of shortint;
var
i,j,k:integer;
```

```
utem:file of ritmus;
teljes_ritmus:array[0..320]of array [0..123] of shortint;
begin
assign(utem,'C:\Szakdolgozat\metes\adatok\res_racs\ritmus1.kep
');
rewrite(utem);
for i:=0 to 320 do
 for k:=0 to 123 do
  teljes_ritmus[i][k]:=round(31*sin((Pi*k/62)-Pi*i/10));
write(utem,teljes_ritmus);
close(utem);
end.
```

A rés mögötti pontok állapotát a rés pontjai (mint hullámcentrumok) által okozott állapotok összege adja. Az így kiszámított állapotot egészre kell kerekíteni, majd hozzáadva 31-et, megkapjuk az adott pont színkódját. Ez rendkívül sok számítást igényelő feladat lenne, amire szimulációt építeni igen nehézkes lenne. (450 Mhz-es számítógépen ez a feladat kb. fél órát vesz igénybe.) Ezért ebben az esetben célszerű a hullámtér minden pontjának állapotát egy teljes periódusra lementeni, egy 124x320x480-as tömbben (res.kep). A megfelelő program részlet:

```
program resek;
uses crt,Windows ;
type
matrix=array[-320..320,-240..240]of integer;
ritmus=array[0..800] of array[0..123]of real;
res=array[0..123]of array[320..640,0..480]of byte;
var
tav:file of matrix;
utem:file of ritmus;
res1:file of res;
kepernyo:array[0..123]of array[320..640,0..480]of real;
kepernyo2:array[0..123]of array[320..640,0..480]of byte;
tavolsag:array[-320..960,-240..720]of integer;
hullamter:array[0..123]of array [-320..960,-240..720]of real;
utemek:array[0..800] of array[0..123]of real;
```

```
i,j,k,l: integer;
procedure kezdoallapot;
var
negyed_tav:matrix;
begin
assign(tav,'tavolsag.kep');
assign(utem,'ritmus2.kep');
assign(res1,'res2.kep');      {a lementett rés neve}
rewrite(res1);
reset(tav);
reset(utem);
read(tav,negyed_tav);
read(utem,utemek);
for i:=-320 to 320 do
 for j:=-240 to 240 do
 begin
 tavolsag[i][j]:= negyed_tav[i][j];
 tavolsag[640-i][j]:= negyed_tav[i][j];
 tavolsag[i][480-j]:= negyed_tav[i][j];
 tavolsag[640-i][480-j]:= negyed_tav[i][j];
 end;

end;
procedure inditas;
begin
for k:=0 to 123 do
 for i:=320 to 640 do
  for j:=-240 to 720 do
  hullamter[k][i][j]:=utemek[tavolsag[i][j]][k];
end;
procedure kepernyoinditas;
begin
for k:=0 to 123 do
 for i:=320 to 640 do
```

```
  for j:=0 to 480 do
  kepernyo[k][i][j]:=0;
end;
begin
kezdoallapot;
inditas;
kepernyoinditas;
for k:=0 to 123 do
 for i:=320 to 640 do
  for j:=0 to 480 do
begin
for l:=0 to 1 do  {ezt kell majd változtatni: résszélesseg/2 }
kepernyo[k][i][j]:=kepernyo[k][i][j]+
hullamter[k][i][j+l]+hullamter[k][i][j-l];
kepernyo2[k][i][j]:=round(kepernyo[k][i][j])+31;
end;
write(res1,kepernyo2);
close(res1);
close(utem);
close(tav);
end.
```

Ezzel a programrészlettel lementettünk 8 különböző esetet: 320, 160, 80, 40, 20, 10, 4, 2 pontokból álló rések által gerjesztett hullámteret. A program, futás közben csak azt olvassa ki a „res.kep” fájlból, hogy a hullámtér vizsgált pontja az aktuális pillanatban milyen állapotban van. Hogy a jelenségek jól láthatóak legyenek, ábrázolni fogjuk az intenzitás-szerű viszonyokat is úgy, hogy a hullámmentes területeket sötétebb, míg az intenzívebb területeket világosabb színnel jelenítjük meg. Azaz a hullámtér aktuális pontjának értékéből kivonunk 31-et, és az eredménynek vesszük az abszolútértékét. Így megkapjuk az adott pont „intenzitását.” A megfelelő program részlet:

```
program res_hullam;
uses
crt,
Windows,
VGFX,
```

```
VGFX_2D,
VGFX_Sprites,
VGFX_Text,
VGFX_win32,
VGFX_Files,
VGFX_Errors,
BigFile2,
SysUtils;
type
ritmus_res=array[0..320] of array[0..123]of shortint;
matrix_res= array [0..123]of array[320..640,0..480]of byte;
var
vw1,vw2 : virtualwindow;
res100 : file of matrix_res;
utem_res:file of ritmus_res;
fel_kepernyo:array [0..123]of array[320..640,0..480]of byte;
kepernyo: array[0..320,0..480]of shortint;
utemek_res: ritmus_res;
i,j,k,n:integer;
procedure kezdoallapot_bal;
begin
assign(utem_res,'ritmus1.kep');
reset(utem_res);
read(utem_res,utemek_res);
close(utem_res);
end;
procedure kezdoallapot_jobb(x:integer);
begin
case x of
 0: begin
assign(res100 ,'C:\Szakdolgozat\metes\adatok\res_racs\resek\re
s320.kep');
    reset(res100);
    read(res100,fel_kepernyo);
```

```
    close(res100);
    end;
 1: begin
assign(res100 ,'C:\Szakdolgozat\metes\adatok\res_racs\resek\re
s160.kep');
    reset(res100);
    read(res100,fel_kepernyo);
    close(res100);
    end;
 2: begin
assign(res100 ,'C:\Szakdolgozat\metes\adatok\res_racs\resek\re
s80.kep');
    reset(res100);
    read(res100,fel_kepernyo);
    close(res100);
    end;
 3: begin
assign(res100 ,'C:\Szakdolgozat\metes\adatok\res_racs\resek\re
s40.kep');
    reset(res100);
    read(res100,fel_kepernyo);
    close(res100);
    end;
 4: begin
assign(res100 ,'C:\Szakdolgozat\metes\adatok\res_racs\resek\re
s20.kep');
    reset(res100);
    read(res100,fel_kepernyo);
    close(res100);
    end;
 5: begin
assign(res100 ,'C:\Szakdolgozat\metes\adatok\res_racs\resek\re
s10.kep');
    reset(res100);
```

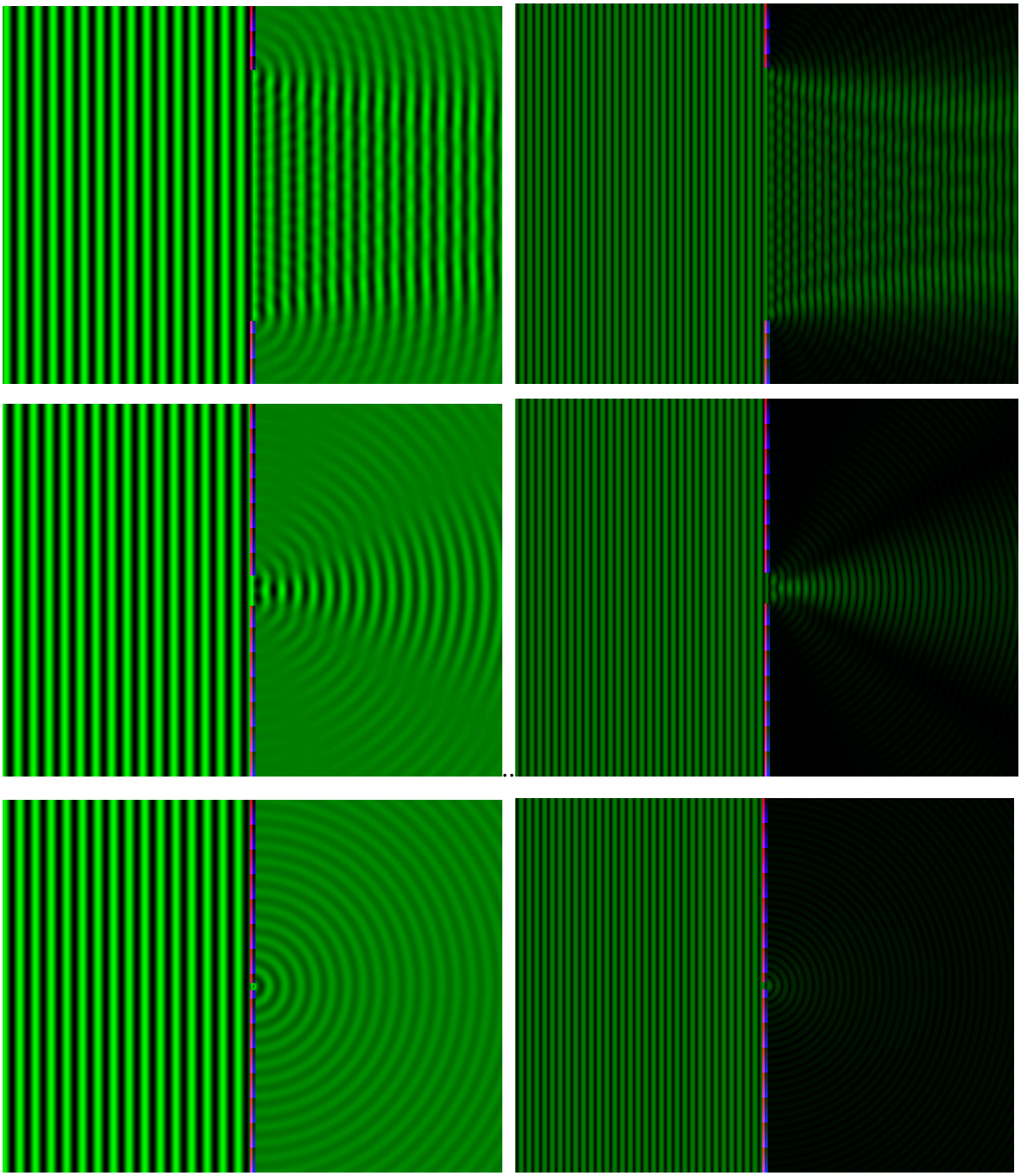
```
    read(res100,fel_kepernyo);
    close(res100);
    end;
 6: begin
assign(res100 ,'C:\Szakdolgozat\metes\adatok\res_racs\resek\re
s4.kep');
    reset(res100);
    read(res100,fel_kepernyo);
    close(res100);
    end;
 7: begin
assign(res100 ,'C:\Szakdolgozat\metes\adatok\res_racs\resek\re
s2.kep');
    reset(res100);
    read(res100,fel_kepernyo);
    close(res100);
    end;
 end;
end;
procedure eltolas;
begin
for i:=0 to 320 do
 for j:=0 to 480 do
 kepernyo[i][j]:=utemek_res[i][k]+31;
end;
procedure  res(x:integer);
var
i,j:integer;
begin
case x of
 0: begin
    for i:=317 to 323 do
      begin
      for j:=0 to 80 do    putpixel(vw1,i,j,i,0,j) ;
```

```
      for j:=400 to 480 do  putpixel(vw1,i,j,i,0,j) ;
      end;
    end;
1: begin
    for i:=317 to 323 do
      begin
      for j:=0 to 160 do  putpixel(vw1,i,j,i,0,j) ;
      for j:=320 to 480 do  putpixel(vw1,i,j,i,0,j) ;
      end;
    end;
2: begin
    for i:=317 to 323 do
      begin
      for j:=0 to 200 do  putpixel(vw1,i,j,i,0,j) ;
      for j:=280 to 480 do  putpixel(vw1,i,j,i,0,j) ;
      end;
    end;
3: begin
    for i:=317 to 323 do
      begin
      for j:=0 to 220 do  putpixel(vw1,i,j,i,0,j) ;
      for j:=260 to 480 do  putpixel(vw1,i,j,i,0,j) ;
      end;
    end;
4: begin
    for i:=317 to 323 do
      begin
      for j:=0 to 230 do  putpixel(vw1,i,j,i,0,j) ;
      for j:=250 to 480 do  putpixel(vw1,i,j,i,0,j) ;
      end;
    end;
5: begin
    for i:=317 to 323 do
      begin
```

```
      for j:=0 to 235 do  putpixel(vw1,i,j,i,0,j) ;
      for j:=245 to 480 do  putpixel(vw1,i,j,i,0,j) ;
      end;
    end;
 6: begin
    for i:=317 to 323 do
      begin
      for j:=0 to 238 do  putpixel(vw1,i,j,i,0,j) ;
      for j:=242 to 480 do  putpixel(vw1,i,j,i,0,j) ;
      end;
    end;
 7: begin
    for i:=317 to 323 do
      begin
      for j:=0 to 239 do  putpixel(vw1,i,j,i,0,j) ;
      for j:=241 to 480 do  putpixel(vw1,i,j,i,0,j) ;
      end;
    end;
 end;
end;
begin
k:=0;
n:=0;
kezdoallapot_bal;
kezdoallapot_jobb(n);
Window_RegisterClass(WIN_Normal);
Window_Main_Handle:= Window_CreateClass('NewTime
v.2.0Win',WIN_Normal);
CheckWMCreate;
init_graph(640,480);
init_vw(vw1,640,480,true);
init_vw(vw2,640,480,true);
REPEAT
flip_vw(vw2.vwoffset,vw1.vwoffset);
```

```
eltolas;
if ktaste[13]then begin n:=((n+1)mod 8);
kezdoallapot_jobb(n);end;
if ktaste[107]then begin n:=((n+7)mod 8);
kezdoallapot_jobb(n);end;
for i:=0 to 639 do
 for j:=0 to 480 do
 begin
 if i<=320 then
  if ktaste[40] then
  putpixel(vw2,i,j,0,abs(kepernyo[i][j]-31),0)
  else
  putpixel(vw2,i,j,0,kepernyo[i][j],0) ;
if i>320  then
 if ktaste[40] then
 putpixel(vw2,i,j,0,abs(fel_kepernyo[k][i][j]-31),0)
 else
 putpixel(vw2,i,j,0,fel_kepernyo[k][i][j],0) ;
 end;
res(n);
k:=(k+5)mod 124;
Flip_DD(vw1.VWOffset);
GETALLMESSAGES;
HANDLEMINIMIZED;
UNTIL ktaste[27];
close(utem_res);
kill_vw(vw1);
kill_vw(vw2);
kill_graph;
end.
```

A program futtatásával létrehozott szimuláció esetében jól látható hogy a rés méretének csökkenésével a rés egyre inkább hullámforrásként viselkedik. Azaz a hullám egyre inkább behatol az „árnyékos” területekre. Ha a rés mérete a hullámhossznál kisebb, akkor ténylegesen úgy viselkedik, mintha egy hullámcentrum lenne. Mint ahogy az alábbi ábrák mutatják.



## Hullám elhajlása rácson

Ha a hullám rácson halad át, akkor azt tapasztaljuk, hogy a rács mögötti hullámtérben hullám mentes és hullámsávok váltakoznak. A jelenség annál inkább szembeötlő, ahogy a rácsállandó mérete közeledik a hullámhossz méretéhez. A jelenség Huygens-Fersnel elvvel úgy magyarázható, hogy a rács pontjaiból kiinduló elemi hullámok a hullámmentes területeken éppen kioltják egymást, a hullámos helyeken, pedig erősítik. A szimuláció során a rácsra egyenes vonal hullámokat küldünk, hogy a rács pontjai azonos fázisban legyenek. A program futásá-

nak felgyorsítása érdekében felhasználjuk a résnél lementett „ritmus1.kep” fájlt. A rács mögötti pontok állapotát a rács pontjai (mint hullámcentrumok) által okozott állapotok összege adja. Az így kiszámított állapotot egészre kell kerekíteni, majd hozzáadva 31-et, megkapjuk az adott pont színkódját. Ahhoz hogy a jelenség jól látható legyen, a rácsot x=100 egyenletű egyenesre helyezzük el. Így a rács mögötti hullámtér nagyobb lesz. Ami megnöveli a szimuláció számolási igényét. (450 Mhz-es számítógépen ez a feladat kb. 3/4 órát vesz igénybe.) Ezért ebben az esetben célszerű a hullámtér minden pontjának állapotát egy teljes periódusra lementeni, egy 124x540x480-as tömbben (racs.kep). A megfelelő program részlet:

```
program resek;
uses crt,Windows ;
type
matrix=array[-320..320,-240..240]of integer;
ritmus=array[0..800]of array[0..123]of real;
racs=array[0..123]of array[100..640,0..480]of byte;
var
tav:file of matrix;
utem:file of ritmus;
racs1:file of racs;
kepernyo:array [0..123]of array[100..640,0..480]of real;
kepernyo2:array [0..123]of array[100..640,0..480]of byte;
tavolsag:array [-320..960,-240..720]of integer;
hullamter:array[0..123]of array [-320..960,-240..720]of real;
utemek:array[0..800]of array[0..123]of real;
m,n,i,j,k,l: integer;
procedure kezdoallapot;
var
negyed_tav:matrix;
begin
assign(tav,'tavolsag.kep');
assign(utem,'ritmus2.kep');
assign(racs1,'racs30.kep'); {ezt kell majd változtatni}
rewrite(racs1);
reset(tav);
reset(utem);
```

```
read(tav,negyed_tav);
read(utem,utemek);
for i:=-320 to 320 do
 for j:=-240 to 240 do
 begin
 tavolsag[i][j]:= negyed_tav[i][j];
 tavolsag[640-i][j]:= negyed_tav[i][j];
 tavolsag[i][480-j]:= negyed_tav[i][j];
 tavolsag[640-i][480-j]:= negyed_tav[i][j];
 end;
end;
procedure inditas;
begin
for k:=0to 123 do
 for i:=100 to 860 do
  for j:=-240 to 720 do
  hullamter[k][i][j]:=utemek[tavolsag[i][j]][k];
end;
procedure kepernyoinditas;
begin
for k:=0 to 123 do
 for i:=100 to 640 do
  for j:=0 to 480 do
  kepernyo[k][i][j]:=0;
  end;
begin
n:=1;
m:=0;
kezdoallapot;
inditas;
kepernyoinditas;
for k:=0 to 123 do
 for i:=100 to 640 do
  for j:=0 to 480 do
```

```
  begin
  for l:=0 to 480 do
   begin  {itt kell majd változtatni: rácsállandó/2 }
   if ((l-135) mod 15)=0 then  n:=-n;
   if (l>135 )and (l<345)and (n>0)
   then  kepernyo[k][i][j]:=kepernyo[k][i][j]+
   hullamter[k][i+221][j+l-240];
   end;
   kepernyo2[k][i][j]:=round(kepernyo[k][i][j])+31;
  end;
write(racs1,kepernyo2);
close(racs1);
close(utem);
close(tav);
end.
```

Ezzel a programrészlettel lementettünk 5 különböző esetet: 140, 84, 60, 42, 30-as rácsállandóval rendelkező rács által gerjesztett hullámteret. A program, futás közben csak azt olvassa ki a „racs.kep” fájlból, hogy a hullámtér vizsgált pontja az aktuális pillanatban milyen állapotban van. Hogy a jelenségek jól láthatóak legyenek, ábrázolni fogjuk az intenzitás-szerű viszonyokat is. A megfelelő program részlet:

```
program hullamok;
uses
crt,
Windows,
VGFX,
VGFX_2D,
VGFX_Sprites,
VGFX_Text,
VGFX_win32,
VGFX_Files,
VGFX_Errors,
BigFile2,
SysUtils;
```

```
type
matrix_racs=array [0..123]of array[100..640,0..480]of byte;
ritmus_res=array[0..320]of array[0..123]of shortint;
var
vw1,vw2 : virtualwindow;
racs100:file of matrix_racs;
utem_res:file of ritmus_res;
fel_kepernyo_racs:array [0..123]of array[100..640,0..480]of
byte;
kepernyo:array[0..320,0..480]of shortint;
utemek_res: ritmus_res;
n,i,j,k:integer;
procedure kezdoallapot_bal;
begin
assign(utem_res,'ritmus1.kep');
reset(utem_res);
read(utem_res,utemek_res);
end;
procedure kezdoallapot_jobb_racs(x:integer);
begin
case x of
 0: begin

assign(racs100 ,'C:\Szakdolgozat\metes\adatok\res_racs\racsok\
racs140.kep');
    reset(racs100);
    read(racs100,fel_kepernyo_racs);
    close(racs100);
    end;
 1: begin

assign(racs100 ,'C:\Szakdolgozat\metes\adatok\res_racs\racsok\
racs84.kep');
    reset(racs100);
```

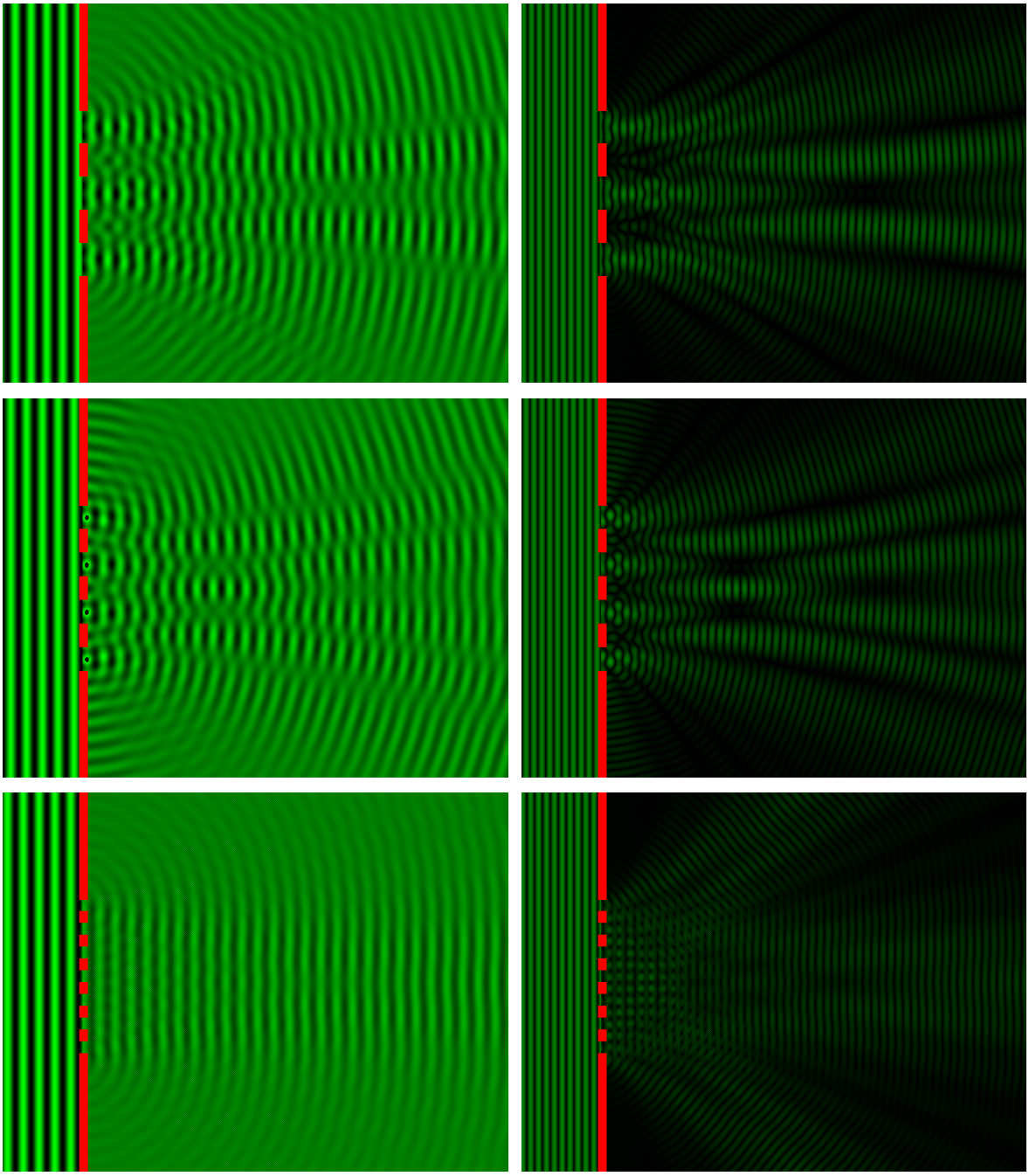
```
    read(racs100,fel_kepernyo_racs);
    close(racs100);
    end;
 2: begin

assign(racs100 ,'C:\Szakdolgozat\metes\adatok\res_racs\racsok\
racs60.kep');
    reset(racs100);
    read(racs100,fel_kepernyo_racs);
    close(racs100);
    end;
 3: begin

assign(racs100 ,'C:\Szakdolgozat\metes\adatok\res_racs\racsok\
racs42.kep');
    reset(racs100);
    read(racs100,fel_kepernyo_racs);
    close(racs100);
    end;
 4: begin

assign(racs100 ,'C:\Szakdolgozat\metes\adatok\res_racs\racsok\
racs30.kep');
    reset(racs100);
    read(racs100,fel_kepernyo_racs);
    close(racs100);
    end;
  end;
end;
procedure eltolas_racs;
begin
for i:=0 to 100 do
 for j:=0 to 480 do
 kepernyo[i][j]:= utemek_res[i][k]+31;
```

```
end;
procedure  racs(x:integer);
var
n,i,j:integer;
begin
n:=1 ;
case x of
 0: begin
    for j:=0 to 480 do
     begin
     if ((j-135)mod 70)=0 then  n:=-n;
     if not((j>135 )and (j<345)and (n>0))then
     for i:=97 to 107 do  putpixel(vw1,i,j,31,0,0) ;
     end;
    end;
 1: begin
    for j:=0 to 480 do
     begin
     if ((j-135 )mod 42)=0 then  n:=-n;
     if not((j>135 )and (j<345)and (n>0))then
     for i:=97 to 107 do  putpixel(vw1,i,j,31,0,0) ;
     end;
    end;
 2: begin
    for j:=0 to 480 do
     begin
     if ((j-135 )mod 30)=0 then  n:=-n;
     if not((j>135 )and (j<345)and (n<0))then
     for i:=97 to 107 do  putpixel(vw1,i,j,31,0,0) ;
     end;
    end;
 3: begin
    for j:=0 to 480 do
     begin
```

```
     if ((j-135)mod 21)=0 then  n:=-n;
     if not((j>135 )and (j<345)and (n<0))then
     for i:=97 to 107 do  putpixel(vw1,i,j,31,0,0) ;
     end;
    end;
 4: begin
    for j:=0 to 480 do
     begin
     if ((j-135)mod 15)=0 then  n:=-n;
     if not((j>135 )and (j<345)and (n>0))then
     for i:=97 to 107 do  putpixel(vw1,i,j,31,0,0) ;
     end;
    end;
end;
end;
begin
k:=0;
n:=0;
kezdoallapot_bal;
kezdoallapot_jobb_racs(n);
Window_RegisterClass(WIN_Normal);
Window_Main_Handle:= Window_CreateClass('NewTime
v.2.0Win',WIN_Normal);
CheckWMCreate;
init_graph(640,480);
init_vw(vw1,640,480,true);
init_vw(vw2,640,480,true);
REPEAT
flip_vw(vw2.vwoffset,vw1.vwoffset);
eltolas_racs;
if ktaste[13]then
  begin
  n:=((n+1)mod 5); kezdoallapot_jobb_racs(n);end;
if ktaste[107]then
```

```
  begin
  n:=((n+7)mod 5); kezdoallapot_jobb_racs(n);end;
for i:=0 to 639 do
 for j:=0 to 480 do
 begin
 if i<=100 then
  if ktaste[40] then
  putpixel(vw2,i,j,0,abs(kepernyo[i][j]-31),0)
  else
  putpixel(vw2,i,j,0,kepernyo[i][j],0) ;
 if i>100  then
  if ktaste[40] then
  putpixel(vw2,i,j,0,abs(fel_kepernyo_racs[k][i][j]-31),0)
  else
  putpixel(vw2,i,j,0,fel_kepernyo_racs[k][i][j],0) ;
end;
racs(n);
k:=(k+5)mod 124;
Flip_DD(vw1.VWOffset);
GETALLMESSAGES;
HANDLEMINIMIZED;
UNTIL ktaste[27];
kill_vw(vw1);
kill_vw(vw2);
kill_graph;
end.
```

A program futtatásával létrehozott szimuláció esetében az intenzitás-szerű képeken jól látható hogy a rácsállandó méretének csökkenésével a hullám egyre inkább különböző irányokban halad. Azaz kialakulnak a hullámmentes és a hullámos sávok. Mint ahogy az alábbi ábrák mutatják:



## A hullámszimuláló program kezelése

A programunk egyszerre fogja kezelni a két hullámcentrum, a rések, a rácsok által létrehozott hullámjelenségeket.  A bal oldali numerikus billentyűzet 0, 1, 2 karakterével lehet váltani az egyes hullámjelenségek között. A klaviatúra jobb oldali numerikus billentyűzetével a hullám terjedésének sebességét szabályozhatjuk. A jobb és bal oldali irányjelző billentyűkkel a két hullámcentrumot tudjuk egymáshoz közelíteni, illetve távolítani. A jobb oldali „Enter”-rel tudjuk a rácsok, illetve rések jellemzőit állítani (rácsállandót, rács méretét). A jobb oldali „+”

billentyűvel tudunk visszalépni. A programot kezelő megfelelő programrészlet:

```
REPEAT
flip_vw(vw2.vwoffset,vw1.vwoffset);
if ktaste[97] then m:=1;
if ktaste[98] then m:=2 ;
if ktaste[99] then m:=3 ;
if ktaste[100] then m:=4  ;
if ktaste[101] then m:=5 ;
if ktaste[102] then m:=6 ;
if ktaste[103] then m:=7 ;
if ktaste[104] then m:=8;
if ktaste[105] then m:=9  ;
if ktaste[96] then m:=10;
if ktaste[48] then x:=0;
if ktaste[49] then x:=1;
if ktaste[50] then x:=2;
case x of
 0:begin
   if ktaste[37] and (v<320)then v:=v+1;
   if ktaste[39] and (v>0)then v:=v-1;
   for i:=0 to 640 do
    for j:=0 to 480 do
    begin
    if ktaste[40] then
    putpixel(vw2,i,j,0,abs(hullamter[k][i+v][j])+
    abs(hullamter[k][i-v][j]),0)
    else
    putpixel(vw2,i,j,0,hullamter[k][i+v][j]+
    hullamter[k][i-v][j]+31,0) ;
    end;
    k:=(k+m)mod 128;
   end;
 1:begin
   eltolas;
```

```
  if ktaste[13]then
  begin
  n:=((n+1)mod 8);
  kezdoallapot_jobb(n);
  end;
  if ktaste[107]then
  begin
  n:=((n+7)mod 8);
  kezdoallapot_jobb(n);
  end;
  for i:=0 to 639 do
   for j:=0 to 480 do
   begin
   if i<=320 then
   if ktaste[40] then
   putpixel(vw2,i,j,0,abs(kepernyo[i][j]-31),0)
   else
   putpixel(vw2,i,j,0,kepernyo[i][j],0) ;
   if i>320 then
    if ktaste[40] then
    putpixel(vw2,i,j,0,abs(fel_kepernyo[k][i][j]-31),0)
    else
    putpixel(vw2,i,j,0,fel_kepernyo[k][i][j],0) ;
   end;
   res(n);
   k:=(k+m)mod 124;
  end;
2:begin
  eltolas_racs;
  if ktaste[13]then
  begin
  n:=((n+1)mod 5);
  kezdoallapot_jobb_racs(n);
  end;
```

```
    if ktaste[107]then
    begin
    n:=((n+7)mod 5);
    kezdoallapot_jobb_racs(n);
    end;
    for i:=0 to 639 do
     for j:=0 to 480 do
     begin
     if i<=100 then
      if ktaste[40] then
      putpixel(vw2,i,j,0,abs(kepernyo[i][j]-31),0)
      else
      putpixel(vw2,i,j,0,kepernyo[i][j],0) ;
     if i>100 then
      if ktaste[40] then
      putpixel(vw2,i,j,0,abs(fel_kepernyo_racs[k][i][j]-31),0)
      else
      putpixel(vw2,i,j,0,fel_kepernyo_racs[k][i][j],0) ;
     end;
     racs(n);
     k:=(k+m)mod 124;
   end;
  end;
Flip_DD(vw1.VWOffset);
GETALLMESSAGES;
HANDLEMINIMIZED;
UNTIL ktaste[27];
```

## Összefoglalás

Tehát szimulációs program megírását alapvetően a következők határozták meg, a hullámtér pontjait külön-külön kell tudni kezelni. Ez egy olyan programozási nyelvet igényel, amely elég gyors grafikával rendelkezik. Nagy mennyiségű számolás, amely nagyon időigényes. A megoldás, hogy előre kiszámoljuk a matematikai számolások eredményét és lementjük őket egy fájlba. Ez viszont a program működésének helyigényét növeli.

## Irodalomjegyzék

Budó Ágoston: Kísérleti fizika I Tankönyv kiadó, Budapest, 1986.

Angster Judit-Arató Éva: Akusztikai példatár, Akadémia kiadó, Budapest, 1986.

http://codexonline.hu/CodeX10/alap/freepascal/InczeAttila/part1.htm (2007 április 18)

http://free-pascal.extra.hu/freepascal.htm (2007 április 18)